Why Digital Signatures?
● Technology for Digital Signatures
● Implementation in
● OpenOffice.org/StarOffice
Future Outlook
● Authentication
● Authorization
● Verify Document Integrity
●
Authentication
A Basic Security Issue is Authentication
●
Authentication is the process of confirming
●
the identity of an entity (a user, a machine, a company)
Securely identify the Author of a Document
●
Authorization
Authentication serves as the basis for
●
Authorization
Specifically, once it knows the identity of a
●
subject, an application may then specify what set of operations that subject may perform
Authorization
Macro Security
●
Decide whether to run a macro or not
–
Decision based on author, not on macro content,
●
because you can't review every macro you receive
Fine Grained Macro Security
–
Macros from different authors can have different
●
access rights to systems resources
Authorization
Digital Rights Management
●
Digital Signatures can be used for simple DRM:
–
Who is allowed to read/modify/print the document?
DRM cannot be enforced with this, only be
–
implemented in specific applications
Verify Document Integrity
Verify that the Document Content was not

●
altered
A document might simply be corrupted because
–
of broken file transfer, but it can also be
manipulated by Intention
Checksums / Hash Values can be used to verify
–
the Content
Technology for Digital Signatures
Hash Values
●
Encryption Algorithms
●
Public Key Certificates
●
Hash Values
One Way Hash Functions
●
Easy to compute, but difficult to reverse
–
Difficult to find two input values which result in
–
same output value
Can be used to calculate Hash Values for the
–
document content, which are much smaller than
the Document Content itself
Hash Values can be stored at a separate place or as
–
part of the Signature
Use same Hash Function to compute Hash Values
–
for a received Document, compare with the saved
Values
Encryption Algorithms
Change Data so it can only be read with the
●
proper Encryption Key
Original text is called “Plain Text”
–
Transformed text is called “Cipher Text”
–
Recovering Plain Text from Cipher Text only
–
possible with correct Key
Encryption Types
●
Symmetric Encryption
–
Asymmetric Encryption
–

Symmetric Encryption
- Same Key is used for Encryption and Decryption
- Also called “Secret Key” Ciphers
- Two communicating parties must share a secret key. This requirement creates some difficulties in key management and key distribution

Improve Crypto Strength
- Symmetric Ciphers can be stacked to improve Crypto strength of the whole system, such as in the case of triple-DES

Asymmetric Encryption
- Pair of Keys, one is used for Encryption, the other for Decryption
- Private Key is only known by the Owner,
- Public Key can be know by everybody and Easily being distributed via Public Channels

Asymmetric Encryption
- In some Asymmetric Systems the Encryption and Decryption is reversible
- This means that one can apply the decryption operation with the private key to the plaintext to get ciphertext, and one can recover the plaintext by applying the encryption operation with the public key to the ciphertext.
- Only the holder of the private key can generate the ciphertext with these systems, so the ciphertext can serve as a digital signature of the plaintext, and anyone with the public key can verify the authenticity of the signature

Public Key Certificates
- Users of public-key applications and systems must be confident that the public key of a subject is genuine
- Public-key certificates are used to establish trust. A public-key certificate is a binding of a public key to a subject, whereby the certificate is digitally signed by the private key of another entity, often called a Certification Authority (CA)

Public Key Certificates
Holds together Information about the Owner
- Name, Email Address, Public Key, Company
  - Name, ...

The standard digital certificate format is
- ITU-T X.509.

An X.509 certificate binds a public key to a
- Distinguished Name

Implementation in
OpenOffice.org/StarOffice
W3C XML DSIG
- OOo Security Framework
- W3C XML DSIG

OpenOffice.org implements Digital
- Signatures following W3C XML DSIG Recommendations

Structure of the Signature itself is XML
- Used Encryption and Hash Algorithms are
  - stored within the Signature

Signature can easily be verified with different
- implementations

W3C XML DSIG
XML Content can be verified via DOM,
- physical representation doesn't matter

Implementation for Binary Data like Pictures
- is also available

OOo Security Framework
Application shouldn't care about Certificate
- Deployment and Management

Use existing Infrastructure
- MS Crypto on Windows
  - Mozilla / NSS on Linux and Solaris
  - OOo Security Framework

LibXMLSec, LibXML2
- OpenOffice.org uses LibXMLSec and

- LibXML2 to create and verify Digital Signatures
- LibXMLSec can handle MS Crypto API and NSS
- The OOo Security Framework is designed to support different implementations, like Java JSR 105/106

Demonstrations

Digital Signatures

- Adding some Signatures
- Broken Signatures
- Demonstrations

Macro Security

- Security Levels
- Trusted Sources and Authors
- Other Security Options

Document Encryption

- Warn if certain Information are saved, signed or printed and have Versions
- Redlining Information
- Notes
- Remove Personal Information on Saving
- Recommend Password Protection on Saving
- Demonstrations

Other Security Options

- Future Outlook

Fine Grained Macro Security

- UNO already offers a mechanism for this
- Needs a lot of changes in UNO API
- Implementations

W3C Encryption using Certificates

- Needs Enhancements to our current framework
  - Workflow and Key Retrieval undefined
  - (Digital Rights Management)